Benchmark tutorial

Prepared to guide towards Benchmark comparisons needed for the LCG EW group.

Preparation

If you have the instructions but not the git repository, clone the files.

I. Running Madgraph

The directory `madgraph` contains several files prepared for `lxplus`: (1) `setup.sh` activates three needed environments (the LCG View, `Madgraph5_aMC@NLO`, fastjet), and (2) `W-P8.dat` is a sample file for the generator `Madgraph5_aMC@NLO` . It describes the decay $W \to \ell\nu j$, where $\ell = e, \mu$.

Run the commands

```
cd madgraph
source setup.sh
mg5_aMC W-P8.dat    2>&1 | tee out-w-p8.log
```

These programs will produce a directory, specified after the keyword `output`, namely, `test-W-P8`. This code produces the hard-process events files.

Remarks:

1. The currently activated version is

```
$ which mg5_aMC
/cvmfs/sft.cern.ch/lcg/releases/LCG_96/MCGenerators/madgraph5amc/2.7.2.atlas3/x8
6_64-centos7-gcc8-opt/bin/mg5_aMC
```

2. The input file `W-P8-lhapdf.dat` requests LHAPDF set:

```
set pdlabel=lhapdf
set lhaid=261000          # NNPDF3
```

However, they are currently not compatible with the `mg5_aMC`, and lead to run-time errors.

3. The current contents of `madgraph/setup.sh`:

```
$ cat setup.sh
source /cvmfs/sft.cern.ch/lcg/views/setupViews.sh LCG_96 x86_64-centos7-gcc8-opt

source
/cvmfs/sft.cern.ch/lcg/releases/LCG_96/MCGenerators/madgraph5amc/2.7.2.atlas3/x8
6_64-centos7-gcc8-opt/madgraph5amcenv-genser.sh
source  /cvmfs/sft.cern.ch/lcg/releases/LCG_96/fastjet/3.3.2/x86_64-centos7-
gcc8-opt/fastjet-env.sh
```

II. Running Pythia8

The directory `pythia8` contains instructions needed to propagate the generated event towards the measurable state.

Remark: Since we will need a valid LHE file with events, lets expand the file created in our previous tutorial. It is located in `madgraph/test-W-P8/Events/run_01/`. The file named `events.lhe.gz` should be decompressed with `gzip -d`.

Testing Pythia8

First, lets us check whether we understand how Pythia8 set-up works. Having activated the LCG environment, lets explicitly activate Pythia8 v.3.02:

```
source /cvmfs/sft.cern.ch/lcg/views/setupViews.sh LCG_96 x86_64-centos7-gcc8-opt

export
PYTHIA8root=/cvmfs/sft.cern.ch/lcg/releases/LCG_96/MCGenerators/pythia8/302/x86_
64-centos7-gcc8-opt
export PYTHIA8DATA=`${PYTHIA8root}/bin/pythia8-config --xmldoc`
```

Those commands are encapsulated in `pythia8/setup.sh`. The default `pythia8-config` points to v.3.01, therefore we explicitly export `PYTHIA8root`.

Since we will run the code based on Pythia8 example `main31.cc`, lets first get its initial version and compile it:

```
mkdir simpleTest1
export src=${PYTHIA8root}/share/Pythia8/examples/
cp $src/main31.cc $src/main31.cmnd $src/Makefile.inc $src/Makefile simpleTest1
cd simpleTest1
make main31
```

This code reads the command file `main31.cmnd`, where the input file `powheg-dijets.lhe` is specified. We can either change the definition of the input file in `main31.cmnd` or simply create a soft link to our previously-created event file, and run the code:

```
ln -s ../../madgraph/test-W-P8/Events/run_01/events.lhe  powheg-dijets.lhe
./main31
```

Our purpose here was to see some meaningful output, containing no error messages.

We could also try to use `jet` events file (166MB) available on `lxplus`. Copy the file, expand it, relink `powheg-dijets.lhe` and run the code (note that this file contains 100k events, so you might want to edit `main31.cmnd` file to limit the processed number of events):

```
cp /afs/cern.ch/user/j/jung/work/public/benchmark/pythia/bin/jet-13TeV-ptj100-
TMDset1-P8-86.lhe.gz  ./
gunzip jet*
rm powheg-dijets.lhe
ln -s jet-13TeV-ptj100-TMDset1-P8-86.lhe powheg-dijets.lhe
```

Preparing Pythia8 executable

We will adapt out code to use the HepMC event package. A Pythia8 example is `main44.cc`. Here we will outline the adaptation of `main31.cc`, but the intention is to use the prepared code, as explained below.

If you started from the initial example, add the following lines to `Makefile` after line, containing definition for `main%`:

```
main%hepmc: $(PYTHIA) main%hepmc.cc
  $(CXX) $@.cc -o $@ $(CXX_COMMON) $(HEPMC2_LIB)
```

Note that the second line shall start with `<tab>` symbol.

Copy the file `main31.cc` to `main31hepmc.cc` and edit this file. We will assume that the environment will have 2 variables set: `PYSEED` (an integer shift to the random seed) and `HEPMCOUT` (name of the file for `HepMC` output).

Code adaptation steps:

a) Add an additional dependency:

```
#include "Pythia8Plugins/HepMC2.h"
```

b) Add the following lines after the comment "Load configuration file":

```
  // Load configuration file
  pythia.readFile("main31hepmc.cmnd");
  pythia.readString("Random:setSeed = on");
  const char * pseed = getenv ("PYSEED");
  if (!pseed) { cout << "env PYSEED is empty" << endl; exit(2); }
  string seed1 = "Random:seed = ";
  string seed2 = pseed ;
  cout << seed1 + seed2 << endl;
  pythia.readString(seed1+seed2);

  // Interface for conversion from Pythia8::Event to HepMC one.
  HepMC::Pythia8ToHepMC ToHepMC;
  // Specify file where HepMC events will be stored.
  const char * outfile = getenv ("HEPMCOUT");
  if (!outfile) { cout << "env HEPMCOUT is empty" << endl; exit(2); }
  HepMC::IO_GenEvent ascii_io(outfile, std::ios::out);
```

c) After `pythia.init()`, add the lines:

```
  double sigmaTotal = 0., errorTotal = 0.;
  double sigmaSample = 0., errorSample = 0.;
  double xs = 0.;
  for (int i=0; i < pythia.info.nProcessesLHEF(); ++i)
      xs += pythia.info.sigmaLHEF(i);
```

d) And the following lines before the "End of event loop":

```
    // Get event weight(s).
    double evtweight         = pythia.info.weight();

    // Do not print zero-weight events.
```

```
    if ( evtweight == 0. ) continue;

    // Construct new empty HepMC event.
    HepMC::GenEvent* hepmcevt = new HepMC::GenEvent();
    // Work with weighted (LHA strategy=-4) events.
    double normhepmc = 1.;
    if (abs(pythia.info.lhaStrategy()) == 4)
      normhepmc = 1. / double(1e9*nEvent);
    // Work with unweighted events.
    else
      normhepmc = xs / double(1e9*nEvent);

    // Set event weight
    //hepmcevt.weights().push_back(evtweight*normhepmc);
    // Fill HepMC event
    ToHepMC.fill_next_event( pythia, hepmcevt );
    // Add the weight of the current event to the cross section.
    sigmaTotal  += evtweight*normhepmc;
    sigmaSample += evtweight*normhepmc;
    errorTotal  += pow2(evtweight*normhepmc);
    errorSample += pow2(evtweight*normhepmc);
    // Report cross section to hepmc
    HepMC::GenCrossSection xsec;
    xsec.set_cross_section( sigmaTotal*1e9, pythia.info.sigmaErr()*1e9 );
    hepmcevt->set_cross_section( xsec );
    // Write the HepMC event to file. Done with it.
    ascii_io << hepmcevt;
    delete hepmcevt;
} // End of event loop.
```

Irrespectively of whether you obtained the modified code `main31hepmc.cc` by doing modifications yourself or taking the source file from the local example directory `SimpleTest2`, you can compile is using

```
make main31hepmc
```

(This will work, if the `Makefile` was updated to link-in libraries defined in `$(HEPMC2_LIB)`.)

We also need the input file `main31hepmc.cmnd`. For testing purposes we could use the default file `main31.cmnd` or the modified file, obtainable from a directory on lxplus:

```
cp /afs/cern.ch/user/j/jung/work/public/benchmark/pythia/bin/mcatnlo-P8.cmnd \
           main31hepmc.cmnd
```

(This file is also available in the `SimpleTest2` directory.) Note that this input file reads events from a file named `pwgevents.lhe`.

The test run on our `Madgraph` example

```
export PYSEED=12314
export HEPMCOUT=hepmc.out
ln -s ../../madgraph/test-W-P8/Events/run_01/events.lhe pwgevents.lhe
./main31hepmc
```

produced `hepmc.out` output file of size of 2MB.

Running Pythia8 and Rivet on a local file

To make a meaningful comparison using Rivet (http://projects.hepforge.org/rivet/, https://twiki.cern.ch/twiki/bin/viewauth/CMS/Rivet), we shall run the executable on the jet event sample.

Create a new subdirectory `tutorial/pythia8/rivetTest3`. Here it is assumed that this directory is parallel to previously created `tutorial/pythia8/simpleTest1`. Copy the code file and compile it, obtain the data file and create a soft-link to it:

```
# don't forget to activate the environment
cd tutorial/pythia8
source setup.sh
# create a new directory for the rivet tutorial
mkdir rivetTest3
cd rivetTest3
# copy previously prepared source file and compile it
cp ../simpleTest2/main31hepmc.c* ../simpleTest2/Makefile*  ./
make main31hepmc
# obtain the jet sample
cp /afs/cern.ch/user/j/jung/work/public/benchmark/pythia/bin/jet-13TeV-ptj100-
TMDset1-P8-86.lhe.gz  ./
gunzip jet*
ln -s jet-13TeV-ptj100-TMDset1-P8-86.lhe  pwgevents.lhe
```

The contents of the current directory shall look like

```
$ ls -la
drwxr-xr-x.  2 andriusj zh      2048 Jul 13 11:50 .
drwxr-xr-x. 10 andriusj zh      2048 Jul 13 11:45 ..
-rw-r--r--.  1 andriusj zh 116493207 Jul 13 11:49 jet-13TeV-ptj100-TMDset1-P8-
86.lhe
-rwxr-xr-x.  1 andriusj zh    138640 Jul 13 11:48 main31hepmc
-rw-r--r--.  1 andriusj zh      5098 Jul 13 11:47 main31hepmc.cc
-rw-r--r--.  1 andriusj zh      4648 Jul 13 11:47 main31hepmc.cmnd
-rw-r--r--.  1 andriusj zh      5398 Jul 13 11:47 Makefile
-rw-r--r--.  1 andriusj zh      3047 Jul 13 11:47 Makefile.inc
lrwxr-xr-x.  1 andriusj zh        34 Jul 13 11:50 pwgevents.lhe -> jet-13TeV-
ptj100-TMDset1-P8-86.lhe
```

We shall also activate Rivet environment:

```
source /cvmfs/sft.cern.ch/lcg/releases/MCGenerators/rivet/3.1.2-aa7e1/x86_64-
centos7-gcc8-opt/rivetenv.sh
```

(This sourcing is included in `setup.sh`, but we remind about this necessity.)

We may want to increase the number of input events defined in `main31hepmc.cmnd`.

We shall also export environment variables `PYSEED` and `HEPMCOUT`:

```
export PYSEED=12314
export HEPMCOUT=hepmc.out
```

Lets run our analysis code and Rivet's analysis on CMS jet data:

```
./main31hepmc
rivet $HEPMCOUT -a CMS_2016_I1459051
```

5000 events produced a file `hepmc.out` of 534MB.

Remark: If `rivet` gives you a message

```
All analyses were incompatible with the first event's beams
Exiting, since this probably wasn't intentional!
```

you forgot to activate Rivet's environment.

The command

```
rivet $HEPMCOUT -a CMS_2016_I1459051
```

shall report that the rivet file `Rivet.yoda` was produced in current directory.

We can run the Rivet code to create the plots:

```
rivet-mkhtml Rivet.yoda
```

It reports that 14 plots are created. (*Note:* If the code seems to be stuck, you forgot to activate the Rivet environment. This might easily happen, if you logout and login again.)

If you are at CERN or have a fast internet connection to lxplus, you can inspect them by launching a browser:

```
firefox rivet-plots/index.html &
```

Running Pythia8 and Rivet on a named pipe

New files are not needed for this tutorial. You may work in `rivetTest3`. The only difference from the previous tutorial step is using a named pipe to transfer information from `main31hepmc` to `rivet`.

Since `$HEPMCOUT` file gets large, as the number of processed event increases, we can take advantage of the named pipe. It can be created only on a special file system:

```
export HEPMCOUT=/tmp/`whoami`-hepmc.out
mkfifo $HEPMCOUT
ls -la $HEPMCOUT
```

Here we added our username to the named pipe to distinguish it from others. The last command is used to verify that the pipe exists (information line shall start with 'p').

When utilising the named pipe, the program that writes to the named pipe and the program that reads from the named pipe have to be run in parallel. Try:

```
./main31hepmc &              # ampersend `&` is very important here
rivet $HEPMCOUT -a CMS_2016_I1459051
```

If we would like to have log files for inspection, we should add redirection of the streams to the output stream (`2>&1`):

```
./main31hepmc   2>&1 > out-main.log    &
rivet $HEPMCOUT -a CMS_2016_I1459051    > out-rivet.log 2>&1
```

If we run the task interactively and would like to see some activity on screen, we can run the second output through stream divider `tee` :

```
./main31hepmc   2>&1 > out-main.log    &
rivet $HEPMCOUT -a CMS_2016_I1459051  2>&1 | tee rivet.lot
```

Running Pythia8 and Rivet on a batch system

To process a large number of events, we shall use the batch system on lxplus. The batch system requires a wrapper script that tells what should be done once the job is launched on the cluster, and the configuration file that tells how to handle the job.

Assuming that this directory will be useful later, when we will do actual analysis, lets create a clone of `rivetTest3` and name it `rivetTest4` . Create the executable `main31hepmc` . Make sure, there is the event file `pwgevents.lhe.gz` (either physically or a link to it).

Since many names are interconnected in the wrapper script and the configuration file, there are some partially adapted examples in `rivetTest4.tmp` directory. The wrapper script prototype is `pythia-aMC-bird.sub` (it assumes the executable `main31hepmc` with its configuration file `main31hepmc.cmnd`) and the configuration file is `condor-mcatnlo.conf` . There is also an auxiliary script `prepare-condor.sh` that adapts the wrapper script to the local directory.

We have already created the executable `main31hepmc` . Copy the other needed files:

```
cp ../rivetTest4.tmp/condor-mcatnlo.conf  .
cp ../rivetTest4.tmp/pythia-aMC-bird.sub .
cp ../rivetTest4.tmp/prepare-condor.sh .
./prepare-condor.sh
```

The last command will adapt `pythia-aMC-bird.sub` file to current directory and create a file `work.sub` that is referenced in `condor-mcatnlo.conf` .

The work directory should look like

```
$ ls -la
total 16346
drwxr-xr-x.  3 andriusj zh     2048 Jul 13 15:12 .
drwxr-xr-x. 11 andriusj zh     2048 Jul 13 13:34 ..
-rw-r--r--.  1 andriusj zh     1375 Jul 13 13:42 condor-mcatnlo.conf
-rw-r--r--.  1 andriusj zh 16564008 Jul 13 15:10 jet-13TeV-ptj100-TMDset1-P8-
86.lhe.gz
-rwxr-xr-x.  1 andriusj zh   138744 Jul 13 15:12 main31hepmc
-rw-r--r--.  1 andriusj zh     5274 Jul 13 13:38 main31hepmc.cc
-rw-r--r--.  1 andriusj zh     4648 Jul 13 13:38 main31hepmc.cmnd
-rw-r--r--.  1 andriusj zh     5398 Jul 13 13:37 Makefile
-rw-r--r--.  1 andriusj zh     3047 Jul 13 13:37 Makefile.inc
drwxr-xr-x.  2 andriusj zh     2048 Jul 13 15:09 out
-rwxr-xr-x.  1 andriusj zh      231 Jul 13 14:57 prepare-condor.sh
```

```
lrwxr-xr-x.  1 andriusj zh       37 Jul 13 15:11 pwgevents.lhe.gz -> jet-13TeV-
ptj100-TMDset1-P8-86.lhe.gz
-rwxr-xr-x.  1 andriusj zh     1388 Jul 13 15:04 pythia-aMC-bird.sub
-rw-r--r--.  1 andriusj zh     1536 Jul 13 15:09 work.sub
```

Submit the job:

```
condor_submit condor-mcatnlo.conf
```

The task status can be checked by

```
condor_q
```

The output will be written to the directory `out`. Including `Rivet.yoda` file.

Since these are the tutorial instructions, no further fine-tuning in names is discussed.

Acknowledgements

This tutorial file was prepared by A.Juodagalvis, who followed the LCG EW tutorial by H.Jung.